1-2 الملف الدليلي

- تعتبر لغة C++ أضخم لغة، وذلك لكبر حجمها وتعدد مكتباتها
- عند التعامل معها من خلال إنشاء برنامج ما يتم التعامل بكتابة الملفات الدليلية والرأسية Header Files تسمي أيضاً بملفات التضمين Include Files
- يتم ذلك في لغة C++ من خلال العبارة التالية :

```
#include <iostream.h>
```

- تعني تحميل الملف الدليلى iostream.h إلى ذاكرة الحاسوب حتى يتسنى للمترجم معرفة أو ترجمة العمليات الخاصة بالإدخال والإخراج
- نلاحظ أن هذا الملف له اسم التمديد h. اختصار لـ header file

سؤال

ماذا تعني العلامتين < > التي تحصر اسم الملف الدليلى **<iostream.h>**؟

تعني أن المترجم يبدأ البحث عن الملف الدليلي في الأساسي الذي يحوي كل**(folder)** الفهرس أو المجلد الملفات الدليلية **include** ويسمي

عند استبدال العلامتين **<>** بعلامتي الاقتباس " " فيبدأ البحث بالفهرس الحالي (الذي يحوى الملف المصدر) ويصبح على النحو التالي

#include "iostream.h"

في كلتا الحالتين فسيتم تنفيذ البرنامج ولكن الاختيار المناسب يسرع للمترجم من خلال تحديد موقع الملف الدليلي

1-3التعليقات Comments

- التعليق هو عبارة عن ملاحظة أو معلومة تكتب في البرنامج لتوضح فهم معين أو فكرة محددة
- عند تنفيذ البرنامج يقوم المترجم بتجاهل هذه التعليقات أثناء عملية التنفيذ، مما يعنى أن أي تعبير تكتبه فهو صحيح حتى لو كان من الناحية اللغوية خطأً.
- تستطيع من ++C لأن هنالك صيغ محدده تتيحها لغة خلالها فقط أن تكتب التعليق وهما صيغتان :-

الصيغة الأولى

وهي أن يكون التعليق محصوراً بين C ورثتها من لغة ألـ العلامات

/* */

<u>ومثال لذلك</u>

```
/*This is my first program to calculate the area of the circle with C++) */
```

هذه الصيغة تجعل استخدام أسلوب التعليقات في ألـ C++ مرن وسهل حيث إنها تتيح كتابة أي تعليق في أي عدد من الأسطر شريطة أن يكون محصوراً بين العلامات /* */

<u>الصيغة الثانية:</u>

أن يكون التعليق مبتدئاً بعلامة // وينتهي بنهاية السطر ولا يتعدى ذلك، ومثال لذلك

```
//This Program Will Display Message
```

أو

```
//This is my first program to
//calculate the area of the circle
```

1-4 المتغيرات Variables

تفق كل لغات البرمجة في الحد الأدنى على أربع أنواع من البيانات هي

- الأعداد الرقمية أي الأعداد الصحيحة مثل 1-,1,5,350
- الأعداد الحقيقية أي التي بها خانة عشرية مثل 0.5, 3.14 ,46.87
- البيانات الرمزية characters ونعني بها كل مفاتيح الطباعة من حروف وعلامات وأرقام (مخزنة كأنها حروف وليست أعداد)
- البيانات المنطقية (Logical) أو (Boolean) وهي التي تأخذ فقط قيمتي خطأ أو صواب

الأعداد العائمة (الحقيقية floating point

تخزن الأعداد الحقيقية في جزأين من الكلمة

- ✓ الجزء الأول يخزن فيـــه الكســـر ويســـمى المانتيســا أو الخانات الكسرية المؤثرة (mantissa)
- ✓ الجزء الثاني فيخزن فيه القوة

والجدول ++C هنالك ثلاثة أنواع من الأرقام العائمة في التالي [illegible] ذلك

اسم النوع	الحجم	القيمة الدنيا	القيمة العظمى
float	4byte	$3.4*10^{-38}$	$3.4*10^{38}$
double	8byte	$1.7*10^{-308}$	$1.7*10^{308}$
long double	10byte	$3.4*10^{-4932}$	$3.4*10^{4932}$

الأعداد الصحيحة (Integer Numbersl

هنالك أنواع مختلفة للإعلان عن الأعداد الصحيحة تختلف في حجم مواضع التخزين المخصصة لكل نوع وفي إمكانية تمثيل الأعداد السالبة

هنالك ثلاثة أنواع من الأعداد الصحيحة في لغة C++

نوع المتغير	حجمه	القيمة الدنيا	القيمة العظمى
char	1 Byte	-2^7	2^7-1
short	2 Bytes	-2^{15}	$2^{15}-1$
Long	4 Bytes	-2^{31}	$2^{31}-1$
int		مثل short في أنظمة 16 bit ومثل long في أنظمة 32 bit	

البيانات الرمزية characters

يتم تمثيل الحروف حسب نظم الترميز المعروفة مثل

- نظام آسكي(ASCII) هو النظام التشفير القياسي الأمريكي للمعلومات
- نظام ابسدك (EBCDIC) هو نظام آي بي أم (IBM) يختلف عن النظام الأول في أنه يبدأ بترميز الحروف الصغيرة ثم الحروف الكبيرة ثم الأرقام أما النظام الأول فيقوم بترميز الأرقام ثم الحروف الكبيرة ثم الحروف الصغيرة

يتم تخزين الأحرف في متغيرات من النوع char في لغة ++C كمايلي:

البيانات المنطقية(Boolean)

يتم تمثيل البيانات المنطقية بثنائية واحدة فقط عندما تكون قيمتها واحد تعنى صحيح أو حقيقية وعندما تكون قيمتها صفر تعني خطا . وهنالك بعض المعاملات التي تطبق علي الحدود البوليانية مثل (AND, OR , NOT , XOR) وتسمي بالمعاملات المنطقية أي تعيد قيمة بوليانية كما في الجدول التالي

A	B	A AND B	A OR B	A XOR B	NOT A
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

نحتاج لتخزين المعلومات عند كتابة أي برنامج بلغة C++ الواردة للبرنامج في ذاكرة الحاسوب تحت عناوين يطلق عليها أسماء المتغيرات وبما أن أنواع المعلومات المراد تخزينها تكون عادة مختلفة مثل القيم الحقيقية أو الصحيحة أو الرمزية فإننا نحتاج أن نعلم المترجم في بداية البرنامج عن أنواع المتغيرات التي نريد استخدامها فمثلاً :

نعرف المتغير بذكر الاسم ونوع البيانات التي يمكن أن يحملها هذا المتغير من أي سلسلة تحتوي على أحرفٍ letter تحتيٍّ أو أرقامٍ أو خطٍ underscore(_) بشرط أن لا يبدأ اسم المتغير برقم

تفرق بين الحروف الأبجدية الصغيرة و C++ أن لغة الكبيرة ،فمثلاً VAR , var, Var تتعامل كمتغيرات مختلفة تماماً.

يمكن تعريف المتغيرات في أي مكان في البرنامج لكن يجب تعريفها قبل استعمالها، كما يمكن تعريف المتغيرات التي تنتمي إلي نفس النوع في سطر واحد، بشرط أن نفصل بينها بفاصلة، والأمثلة التالية توضح نماذج لتبيان كيفية تعريف المتغيرات أو الإعلان عنها.

```
int a,b,c;

short int i;

short     i;   // same as "short int

double f,g;

float k,l;
```

1-5 عبارتي الإدخال والإخراج البيانات

لكتابة (C out تقرأ)cout عبارة C++ تستخدم لغة لقراءة (C in تقرأ) cin المعلومات علي الشاشة وعبارة المعلومات من لوحة المفاتيح. فمثلاً العبارة التالية

```
cin>>var1;
```

تخزن الرقم الذي يكتبه المستخدم من لوحة المفاتيح في متغير يسمي var1 حيث يمثل الكائن cin لوحة المفاتيح

يمكن استخدام عامل الحصول عدة مرات في نفس العبارة مثلاً

```
cin>>var1>>var2>>var3;
```

1-6 الدالة main()

- تعتبر الدوال من أهم مقومات البرنامج في لغة C++ ونجد أن البرنامج يمكن أن يتألف من دالة واحدة أو أكثر وانه لابد لكل دالة من اسم يدل عليها عند استخدامها واستدعائها .

- فإن الدالة main() لها خصوصيتها في لغة C++ المترجم يبحث عن هذه الدالة أولاً لتنفيذها، أما إذا لم تكن موجودة فستظهر رسالة خطأ Error Message توضح ذلك.

- فهي تعني أن هذا main() الأقواس التي تلي الدالة الاسم هو اسم لدالة فبدون هذه الأقواس يترجمها المترجم على أساس إنها اسم لمتغير، فلهذا تعتبر هذه الأقواس مهمة جداً

1-6 الـدالـة main()

- الدالـة main() غالباً أو الدوال الأخرى لها قيمـة راجعـة هي حقيقة قيمة الدالة فيتم إرجاع هذه القيمة من خلال الأمر return، أمـا الدوال التـي ليـس لهـا قيمـة راجعـة أو قبل اسم الدالة نكتب كلمة 0 return فنستخدم void التي تعني أن هذه الدالة ليس لها قيمة راجعة
- الأمثلة التالية توضح نماذج لتبين كيفية تعريف الدالة main().

```
main()
{
    . . . . .
    . . . . .
    . . . . .
```

```
void
main()
{
. . . . .
. . . . .
. . . . .
}
```

```
int main()
{
. . . . .
. . . . .
return 0
}
```

Omer Ahmed Saeed

مثـــال (1-1)

```
//This Program Will Display Message
#include <iostream.h>
main()
{
   cout << "welcome to C++ " ;
}
```

إذا كان محرر لغـة ++C يعمـل فـي بيئـة Dos (نظام التشغيل) سوف يقوم الحاسوب بتنفيذ البرنامج ويعود سريعاً للمحرر

معلومة

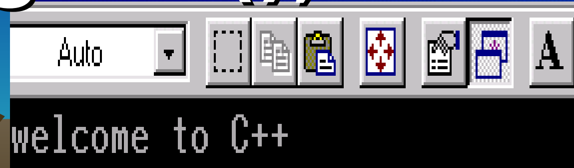
- تتكون برامج لغة ++C من عبارات أو جمل وتنتهي كل عبارة بالفاصلة المنقوطة ";".
- تتكون الأسماء في لغة ++C بأي طول نريده من الرموز شريطة أن يبدأ الاسم بحرف من A-Z أو a-z أو "_"، لكن المترجم يكتفي فقط بأول 63 رمزاً.

ولكـن إذا أردت تثـبيت المخرجات علـي الشاشـة حتـى يتسـنى لـك مشاهدتهـا مـا عليـك إلا إضافـة إلـي نهايـة البرنـــامج وهـي تحتاج **getch()** عبارة إلى الملف الدليلى **#include <conio.h>**

المثال التالي يوضح هذا المفهوم.

مثـــال (1-2)

```
#include <iostream.h>
#include <conio.h >
main()
{
  cout << "welcome to C++ " ;
  getch();
}
```



إخراج البرنامج

7-1 الثوابت

قد نحتاج أن نستخدم قيماً ثابتةً عند كتابة برامجنا ، وهذه القيمـة ثابتـة أينمـا اسـتخدمت ضمـن الـبرنامج ولا يجوز تعديلها (وإلا سوف نحصل علي رسالة خطأ من المترجم) فمثلاً عنـد حسـاب معادلـة الدائرة فإننـا نحتاج إلـى قيمـة وهي قيمة ثابتة ولتعريف الثوابت في لغة C++ نستخدم الصيغة التالية :

const ConstType <ConstName> = <value>;

حيث أن:

- **ConstType** يـمثل نـوع الثابت **int**
- **ConstName**

Omer Ahmed Saeed

مثال لذلك

```
const float pi   = 3.14;
const int maxint  = 12345;
const int x  = 10;
const char ch='a';
```

يجوز أن نعلن عن أكثر من ثابت إذا كانوا من نفس النوع بعبارة const واحدة مثال لذلك

```
const
      float pi = 3.14 ,
             x  = 1000.5;
```

ولا يجوز أن نعلن عن أكثر من ثابت بعبارة const واحدة إذا كانوا من أنواع مختلفة والمثال التالي

```
const
      float pi  = 3.14 ,
      int     x  = 10;
```

ينتج عنه خطأ، وذلك لأن جملة الثابت لم تنتهي بعلامة ";" وعند تصحيح هذا الخطأ كما في الشكل التالي

```
const
      float pi  = 3.14 ;
      int     x  = 10;
```

يعمل البرنامج بصورة صحيحة ولكن يعتبر أن x متغير عادي وليس ثابت وبالتالي يسمح البرنامج بتعديله،

طريقة أخري للتصريح عن الثوابت ورثتها ++C تعتمد لغة وذلك #DEFINE ويتم ذلك من خلال الموجه C من لغة وفقاً للصيغة التالية

#DEFINE ConstName Value;

مثال لذلك

```
#DEFINE float pi   = 3.14;
#DEFINE int maxint  = 12345;
#DEFINE int x  = 10;
#DEFINE char ch=’a’;
```

1-8 الفراغات البيضاء White Space

يتخطى المترجم كل الفراغات الموجودة بالبرنامج أي يمكن أن تتجزأ العبارات إلى عدة اسطر ويمكن أن نكتب عدد من العبارات في سطر واحد، نستفيد من هذه الميزة عند تنظيم برنامجنا. إلا أن هنالك حالات لا يمكن استخدام الفراغات بصورة مطلقة وهي

- الملفات الدليلية (Include Files) التي ينبغي ان تكون في سطر واحد.
- الثوابت الحرفية التي لها طول محدد وهذه الفراغات تحسب ضمن الطول المصرح به في الثوابت الحرفية.
- التعليقات التي تستخدم النمط الثاني الذي ينتهي بنهاية السطر.

وعليه فان هذه الجملة

```
const float Pi = 3.1415926;
```

يمكن أن تكتب علي النحو التالي

```
const
float
Pi
= 3.1415926 ;
```

جملة واحدة تتجزأ إلى عدد من الجمل

1-9 سلاسل الهروب Escape Sequence

سلاسل الهروب هي عبارة عن رموز تؤكل لها مهام محدده ضمن البرنامج، وتضم لغة ++C عدد من سلاسل هروب كما هو مبين في الجدول التالي

الوظيفة	Escape Sequence
سطر جديد (النزول إلى السطر التالي)	n\
لترك مسافة طولها حقل واحد	t\
لاستخدام علامة \	\\
لاستخدام علامة '	\'
لاستخدام علامة "	\"

من أكثر السلاسل "t\" و "n\"تعتبر سلاسل الهروب استخداماً

1-10 المؤثرات Manipulators

المؤثرات هي عبارة عن أدوات تستخدم مع علامة >> من اجل التأثير علي شكل المخرجات والجدول التالي يوضح بعض المؤثرات المستخدمة في لغة C++.

endl	**End Line**	سطر جديد (النزول إلى السطر التالي) أي له نفس تأثير سلسلة الهروب **\n**
setw(N)	**Set Width**	لترك مسافة كافية لإظهار قيمة معينة من خلال تحديد عرض الحقل المناسب و N عبارة عن عدد صحيح يمثل عرض الحقل المطلوب ويحتاج هذا المؤثر إلى الملف الدليلي **iomanip.h**

فمثلاً إخراج العبارة التالية

```
cout<<"Ahmed"<<endl<<"Khalid”<<endl;
```

يكون علي النحو التالي

```
Ahmed
Khalid
```

وإخراج العبارة التالية

```
cout<<"Ahmed"<<setw(20)<<"Khalid”<<endl;
```

يكون علي النحو التالي

```
Ahmed                    Khalid
```

1-11 العمليات الحسابية وعمليات المقارنة وأولوياتها

تنفذ معظم البرامج التي نستخدمها عمليات حسابية وعمليات مقارنة ويتحدد ذلك من طبيعة المشكلة أو المسالة المراد برمجتها.

أولاً : العمليات الحسابية

اسم العملية	الرمز الحسابي	التعبير الجبري	التعبير بلغة ++C
الجمع	+	a+b	a+b
الطرح	-	a-b	a-b
الضرب	*	ab	a*b
القسمة	/	$a \div b \; or \; \frac{a}{b}$	a/b
باقي القسمة	%	A mod b	a % b

يبين الجدول العمليات الحسابية المألوفة بالإضافة إلى العامل الخامس وهو عامل باقي القسمة الذي يسمى المعامل modulus.

ثانياً : عمليات المقارنة

تجري عمليات المقارنة بين قيمتين وتؤدي إلى نتيجة صحيحة أو خطأ وفقاً لما كانت علية عملية المقارنة، الجدول التالي يبين ذلك.

الرمز	المعنى	مثال لذلك
==	يساوي	D==C
!=	لا يساوي	C!=D
<	اكبر من	C>D
>	اصغر من	C<D
=<	اكبر من أو يساوي	C>=D
=>	اصفر من أو يساوي	C<=D

يحدث خطأ قواعدي عندما نضع فراغات بين الرموز المعبرة عن العمليات التالية: <=, >=, ==, != أو عندما نكتبها بصورة معكوسة مثل => بدلاً عن =<

ثالثاً : أولويات العمليات

يقوم مترجم اللغة بتطبيق القواعد الأولوية المستخدمة في الجبر في العمليات الحسابية وعمليات المقارنة وذلك وفقاً الأتي :

- يتم حساب العمليات الموجودة داخل الأقواس أولاً، وإذا كانت الأقواس متداخلة يتم حساب الأقواس المتداخلة أولاً وإذا كانت الأقواس متراصة يتم حسابها من اليسار إلى اليمين فمثلاً يتم في التعبير التالي

```
cout<<(((3+4)%(21/6))+2);
```

- يتم حساب الضرب والقسمة وباقي القسمة ثانياً ولهم نفس الأولوية ويجري حساب هذه العمليات من اليسار إلى اليمين . فمثلاً يتم في التعبير التالي

يــبين الجدول التالــي أولويات العمليات الحســابية وعمليات المقارنة.

العمليات	المعنى	التجميعية
( )	الأقواس	من اليسار إلى اليمين
%,/,*	الضرب،القسمة،باقي القسمة	من اليسار إلى اليمين
-,+	الجمع ، الطرح	من اليسار إلى اليمين
<<,>>	إدخال، إخراج	من اليسار إلى اليمين
=>,>,=<,<	المقارنة	من اليسار إلى اليمين
=!,==	المساواة	من اليسار إلى

1-12 الزيادة والنقصان Increment and Decrement

الزيادة أو النقصان تعني إضافة أو طرح واحد من المتغير علي الترتيب، وكما هو معلوم يمكن أن نعبر عن هذا المفهوم من خلال التعبيرين التاليين :

$a \div b \, or \, \frac{a}{b}$

```
x=x+1 ;
x=x-1 ;
```

تتيح لغة C++ طرقاً أخرى للتعبير عن الزيادة أو النقصان هي

```
x++ ;  // (زيادة بعدية)
++x ;  // (زيادة قبلية)
```

بينما التعبير التالي x=x+1 وهي تعادل أو تكافئ التعبير x=x-1

يعادل أو يكافئ التعابير التالية

```
x-- ;  // (نقصان بعدي)
--x ;  // (نقصان قبلي)
```

ونعني بالزيادة أو النقصان القبلي إتمام عملية الزيادة أو النقصان أولا ومن ثم تنفيذ العملية المطلوبة سواءً كانت طباعة أو غيرها من العمليات ، بينما الزيادة أو النقصان البعدي يقتضي إتمام العملية أولاً ثم إجراء الزيادة أو النقصان ثانياً.

مثال يوضح مفهوم الزيادة القبلية

```
x=5;
cout<<++x;  //output :6
```

نجد أن إخراج هذه الشفرة هو 6 لان عملية الزيادة تتم أولاً ثم إجراء عملية طباعة قيمة المتغير ثانياً، ولكن في المثال التالي:

```
x=5;
cout<<x++;  //output :5
```

تتم عملية طباعة قيمة المتغير وهي 5 أولاً ثم من

سؤال

هل تدعم لغة C++ مفهوم الضرب البعدي والقبلي ؟
أي هل يمكن أن نعبر عن الإجراء x=x*1; بأحد الطريقين :

- ✓ x** ; // (ضرب بعدي)
- ✓ **x ; // (ضرب قبلي)

الإجابة علي السؤال هي النفي،

x هي x ليس لها معني ، قيمة **x لان عبارة نفسها لان الواحد هو عنصر أو قيمة محايدة للضرب وكذلك القسمة ، وبالتالي فان لغة C ++ لا تدعم هذا المفهوم .

سؤال

هل تدعم لغة C++ مفهوم الزيادة أو النقصان بقيمة غير الواحد ؟ مثلاً هل يمكن أن نعبر عن الإجراء x=x+2; بأحد الطريقين :

X++++ ; ++++X ;

الإجابة علي السؤال هي النفي،

لان طريقة الزيادة أ و النقصان جاءت لتقليل عملية التكرار تخيل إذا كانت C++ تدعم هذا والكتابة في لغة C++، بالصورة التالية x=x+4; فأننا نعبر عن المفهوم الإجراء x++++++; لغة C++ لكن لحسن الحظ تمتلك طرق من اجل الاختصار وتقليل التكرار فمثلاً نعبر عن الإجراء x=x+10; بالصورة التالية x+=10; وبصورة عامة نعبر عن الإجراء x=x+a; بالصورة التالية x+=a; وينطبق هذا الاختصار علي كافة العمليات الأخرى مثل الطرح والقسمة والضرب وباقي القسمة وذلك من خلال الأتي:

نعبر عن الإجراء x=x-a; بالصورة التالية x-=a;

نعبر عن الإجراء x=x*a; بالصورة التالية x*=a;

نعبر عن الإجراء x=x/a; بالصورة التالية x/=a;

نعبر عن الإجراء x=x%a; بالصورة التالية x%=a;

(مثـــال 1-5)

أدرس البرنامج التالي جيداً ووضح إخراجه
البرنامج يوضح تأثير الزيادة القبلية والبعدية للمتغير **x**

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int x=10;
    cout<<x<<endl;
    cout<<++x<<endl;
    cout<<x<<endl;
    cout<<x++<<endl;
    cout<<x<<endl;
    getch();
}
```




```
hh
10
11
11
11
12
Press any key to continue_
```

مثـــال (6-1)

وضح إخراج الـبرنامج

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int x=10;
    cout<<x<<endl;
    x+=5;
    cout<<x<<endl;
    x-=3;
    cout<<x<<endl;
    x/=4;
    cout<<x<<endl;
    x*=7;
    cout<<x<<endl;
    x%=6;
    cout<<x<<endl;;
    getch();
}
```



hh

Auto

```
10
15
12
3
21
3
Press any key to continue
```

(مثال 7-1)

أكتب برنامجاً بلغة C++ يقوم بحساب مساحة الدائرة التي نصف قطرها r علماً بأن مساحة الدائرة تحسب وفقاً للمعادلة $area = \pi r^2$.

```
#include <iostream.h>
#include <conio.h>
const float pi=3.14;
void main()
{
        float Area , r;
        cout<<"\nEnter the radi
        cin>>r;
        Area= pi*r*r;
        cout<<"Area ="<<Area;
  getch();
}
```



الإخراج

C:\BC5\BIN\NONAME00.exe

```
Enter the radius : 12

Area = 452.16
```

(مثال 1-8)

يقوم بحساب مساحة أي C++ أكتب برنامجاً بلغة
علماً بأن b وقاعدته a مثلث قائم الزاوية ارتفاعه
مساحة المثلث تحسب وفقاً للمعادلة التالية

$$area = \frac{1}{2} ab$$

```
#include <iostream.h>
#include <conio.h>
void main()
{
	float Area, a, b;
	cout<<"\nEnter the value of a :";
	cin>>a;
	cout<<"\nEnter the value of b : ";
	cin>>b;
	Area= a*b/2;
	cout<<"Area = "<<Area;
	getch();
}
```



C:\BC5\BIN\NONAME00.exe

```
Enter the value of a : 6

Enter the value of b : 9
Area = 27_
```

يقوم بحساب مساحة أي C++ أكتب برنامجاً بلغة

علماً بأن مساحة المثلث حسب a,b,c مثلث أضلاعه

وفقاً للمعادلة التالية

$$s=\frac{a+b+c}{2}$$

$$area=\sqrt{s(s-a)(s-b)(s-c)}$$

حيث

```
#include <iostream.h>
#include <conio.h>
void main()
{
        float Area, a, b, c, s;
        cout<<"\nEnter the value of a : ";
        cin>>a;
        cout<<"\nEnter the value of b : ";
        cin>>b;
        cout<<"\nEnter the value of c : ";
        cin>>c;
        s= (a + b + c)/2;
        Area= sqrt(s*(s-a)*(s-b)*(s-c));
        cout<<"Area = "<<Area;
        getch();
}
```

الإخراج



```
C:\BC5\BIN\NONAME00.exe

Enter the value of a : 6

Enter the value of b : 8

Enter the value of c : 10
Area = 24
```

استخدمنا احد الدوال المكتبية الجاهزة في اللغة التي تقوم بحساب الجزر **sqrt(n)** وهي الدالة التربيعي للعدد وسوف يأتي شرح الدوال المكتبية الجاهزة من خلال دراسة الدوال في هذا الكتاب بإذن الله تعالي وتحتاج هذه الدالة إلي الملف الدليلي **<math.h>**

(مثـــــال 1-10)

أكتب إخراج الـبرنامج الـتالي

```
#include <iostream.h>
#include <conio.h>
void main()
{
        cout<<"*\n**\n***\n****\n*****";
  getch();
}
```




```
C:\BC5\BIN\NONAME04.exe
*
**
***
****
*****
```